Routing and Advanced Display Technologies within STOMPM

Ranjeev Mittu*and Jeffrey K. Uhlmann
Naval Research Laboratory
4555 Overlook Avenue
Washington, DC 20375-5337

Justin McCune
ITT Systems and Sciences Corporation
2560 Huntington Avenue
Alexandria, VA 22303-1404

ABSTRACT

This paper will discuss research conducted at the Naval Research Laboratory (NRL) in the area of automated routing, advanced 3D displays and novel interface techniques for interacting with those displays. This research has culminated in the development of the STrike Optimized Mission Planning Module (STOMPM). The STOMPM testbed incorporates new technologies/results in the aforementioned areas to address the deficiencies in current systems and advance the state of the art in military planning systems.

Keywords: Vehicle Routing, Visualization, Virtual Reality, Mission Planning, Decision Aids

2 INTRODUCTION

In this paper we describe research and development work conducted at the Naval Research Laboratory (NRL) relating to automated routing. In particular, we have examined a variety of battlefield decision support problems associated with the routing of ground and air assets. The culmination of this work is the STrike Optimized Mission Planning Module (STOMPM), which is a flexible research testbed developed mainly for evaluating routing algorithms. It provides capabilities for creating and visualizing simple routing scenarios, defining arbitrary cost functions (e.g., based on models of enemy threats, terrain and environmental conditions, etc.), and computing and assessing auto-generated routes. Although STOMPM development has emphasized the routing of strike aircraft, it leverages much of our previous work on submarine and ground troop mobility applications.

3 BACKGROUND

In the early 1990s, NRL developed a novel battlegroup vulnerability assessment tool [1] based on optimal routing technology. Specifically, given a set of sensor platforms providing well-modeled sensor coverage, the

*Corresponding Author: (202) 404-8716, mittu@ait.nrl.navy.mil

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **APR 1998**	2. REPORT TYPE	3. DATES COVERED **00-00-1998 to 00-00-1998**
4. TITLE AND SUBTITLE **Routing and Advanced Display Technologies within STOMPM**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Research Laboratory,4555 Overlook Ave. SW,Washington,DC,20375**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		

14. ABSTRACT

This paper will discuss research conducted at the Naval Research Laboratory (NRL) in the area of automated routing, advanced 3D displays and novel interface techniques for interacting with those displays. This research has culminated in the development of the STrike Optimized Mission Planning Module (STOMPM). The STOMPM testbed incorporates new technologies/results in the aforementioned areas to address the deficiencies in current systems and advance the state of the art in military planning systems.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**12**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

position of a high valued unit (HVU) or defended asset, and a rectilinear area-of-interest grid, the tool assists the user in identifying the avenues of approach from which the HVU is most vulnerable to attack (Figure 1). This is accomplished by using Dijkstra's algorithm to compute the least cost path from the HVU to every grid cell, which takes $O(N \log N)$ time for N grid cells. Then a display is generated using a heat scale to color each grid cell based on the cost of the best path from that cell to the HVU. Such a display has proven to be extremely natural and intuitive for users to quickly identify vulnerable avenues of approach.

As a specific example, the vulnerability assessment window in Figure 1 depicts the vulnerability of a naval carrier group to submarine attack. The color at each point reflects (using a heat scale) the probability that a submarine at that point could wage a successful attack against the aircraft carrier (the HVU). This probability is computed by determining the avenue of approach that is least likely to produce detection by the carrier group. The white region corresponds to the position of the carrier, and the seven dark dimples correspond to other members of the carrier group having anti-submarine warfare (ASW) resources. The region around the carrier is white-hot because a submarine reaching that proximity represents a substantial threat. Any position from which a submarine must travel through the detection region of one of the carrier group's ASW assets will be blue-cold because it is likely to be detected and deterred.

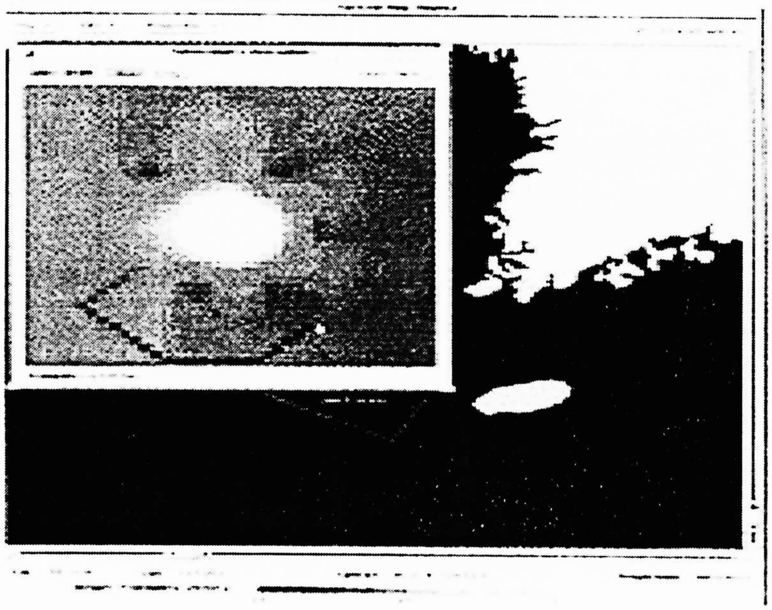

Figure 1: The Vulnerability Assessment Tool. The user clicks on a starting point and the tool generates an optimal path from that point to the HVU

This vulnerability assessment tool was examined for both battlegroup and ground asset defense applications, and we recognized that the combination of optimal routing information and high quality graphical display technology was worth examining in greater generality. Sponsors at the Office of Naval Research (ONR) agreed and funded us to compose a white paper [2] on strike technology deficiencies to identify possible applications of optimal routing. We solicited inputs from Naval strike experts regarding what they believed to be the most significant deficiencies. From their responses, along with our own comprehensive survey of existing strike-related decision support systems, we were able to conclude that indeed most of the enumerated deficiencies could be at least partially addressed using recent advances in optimal routing algorithms and advanced display devices.

Based on the findings of the white paper, we were directed to focus efforts on addressing problems associated with force level routing. Specifically, we were to examine how multiple platforms could be jointly routed to better achieve overall objectives, i.e., better than could be achieved using traditional approaches in which each platform is routed independently. Because of the enormous variety of practical considerations involved in any realistic planning situation, it is clearly impossible to completely automate the process to eliminate the need for human oversight and refinement. Therefore it was also necessary to examine human interface issues involved in facilitating the incorporation of human expertise into the process. This entailed the consideration of how to display the auto-generated routes with respect to environmental and enemy threats, and how to allow the user to easily manipulate and refine the routes to accommodate factors beyond the scope of the models used in the optimization process.

The first step of the project was to define the requirements for a testbed in which both routing and visualization technologies could be developed, tested, and evaluated. This requirements definition led to the development of our primary testbed environment, the STrike Optimized Mission Planning Module (STOMPM). In subsequent sections we will describe in detail the STOMPM system (both version 1.0 and 2.0), but we begin by first describing some of the algorithmic problems associated with routing, from which we move on to graphical user interface (GUI) technologies.

4 PATH OPTIMIZATION TECHNOLOGY

The force level planning problem consists of determining how best to use available assets and resources (e.g., aircraft and weapons) to achieve various strategic objectives (e.g., destroying an enemy power grid). Whereas mission level planners concentrate on a single objective, force level planners must consider the tradeoffs of multiple potential mission objectives. Events in Desert Shield/Storm revealed the complexities of force level planning for Joint Service operations, and it became clear that the process must be automated as much as possible to ensure rapid response capability. However, it was also clear that the experience and expertise of human planners could not be replaced. Thus, the role of computer automation is to provide:

- SPEED - the computer can analyze thousands of scenarios in great depth.
- CONSISTENCY - the computer can ensure that assets are applied according to a consistent criterion. This is the key to getting the most from available resources.

In a large-scale problem it may be impossible for a human to grasp the intricacies of global optimality, but a human is ideally suited to recognize when components of a plan are intolerable for reasons beyond the information available to the computer.

One example of a problem that is difficult for human planners is joint routing [3]. In practice (and doctrine) the force level problem is decomposed into mission level routing problems involving only a small number of assets, targets, and threats. The problem is to determine which of the myriad of possible missions best satisfies overall strategic objectives. In other words, the goal is to find a set of mission level routes which *jointly* maximize the difference between the total payoff achieved minus the total cost incurred by the missions. Some characteristics of jointly optimized strike plans are:

1. The best plan is not generally the one composed of the highest payoff individual missions.
2. It may not even include a strike against the highest valued target if the costs are too high.
3. The best plan will often include negative payoff suppression missions that decrease costs for other missions, thereby increasing the total payoff of the plan.

Unfortunately, research performed in collaboration with the University of New Mexico-Albuquerque resulted in the negative conclusion that almost any realistic formulation of the joint routing problem is provably intractable [4]. Thus, optimality can only be achieved using exhaustive search methods for very small scenarios while suboptimal heuristic approaches are required for larger ones. We were only able to develop efficient and optimal solutions for highly restricted formulations of the joint routing problem.

In the process of developing various approaches for joint routing, we discovered serious deficiencies in the technology being applied for single mission routing. Specifically, we discovered that existing mission routers were unable to adequately accommodate many types of practical constraints such as, e.g., limits on speed, turn radius, and fuel consumption [5]. Most auto-routing systems either used crude heuristics, which either produce poor routes or fail to find feasible routes at all, or require human refinement to enforce constraints, which in some cases results in a complete abandonment of the auto-generated route. We were able to derive new theory and algorithms [6] that are computationally as efficient as the heuristic methods, yet yield optimal solutions which respect the constraints.

5 INFORMATION PRESENTATION AND MANIPULATION

For the purposes of this paper, Information Presentation and Manipulation will encompass several sub-areas. These subareas include visualization, advanced displays, and interaction capabilities. Visualization allows one to visually inspect complex relationships in the environment being modeled. Advanced display technology allows the application to go beyond the use of a standard computer monitor for displaying the visual information. The interaction capabilities allow one to naturally and easily interact with and manipulate the objects contained in the environment.

The next section will discuss a visualization technology that has been investigated, namely the use of stereographics for the inspection of the 3D strike information. It will also discuss the development of a 3D model for a phenomenon in strike warfare know as Radar Terrain Masking (RTM), and the advantage of viewing this model, as well as other 3D information, in stereo mode. When terrain obscures the view of a radar in a given direction, we say that the terrain has "masked" the view of the radar. The following section will discuss an advanced display technology being utilized for the visualization of the 3D strike information, namely the virtual reality responsive workbench. The advantages of using the workbench will be presented. Lastly, we will discuss novel interaction technologies that have been developed within STOMPM for the workbench.

5.1 Visualization techniques

Stereographics [7 8] provides a true 3D representation of an environment by providing two images to the eyes in sequential order, one for the left and one for the right. This allow the user to perceive depth on a two dimensional monitor. Implementing the stereo effect in software is not very difficult and works as follows: produce two images of the scene and double the monitor's refresh rate. One of these images is for the right eye while the other is for the left eye. The user can then wear Liquid Crystal Display (LCD) shutter glasses to view the image in stereo. The shutter glasses work by showing the left eye the image intended for the left eye and the right eye the image intended for the right eye, in alternating sequence (i.e., the shutters in the glasses open and close in synchronization with the monitors refresh rate - the synchronization signal is sent to the glasses from an emitter). The overall effect to the user is a view which more closely resembles 3D - the stereo image can be either projected in front of, or behind, the computer screen, by adjusting a parameter in the software that controls the distance from the eyes to the image convergence point.

There are several items worth mentioning about the use of stereo. As was already mentioned, the stereo image can be projected in front of, or behind, the computer screen by adjusting a certain parameter in the software. When one sets the parameter such that the image is projected in front of the screen, the eyes can get confused by a floating image in front of the screen, which when seen in comparison to the edges of the window/display, appear underneath the window/display (edge effects). Zooming or panning effects can further magnify the "edge effect" phenomenon. Therefore, it is important to have the entire scene visible when one wishes to project the image in front of the computer screen. However, by having the entire scene visible on the screen, it may be impossible to view the important details associated with the scenario. In many instances, it is desirable to project the image behind the screen. What we have noticed is that in our particular application, when the maximum height of the viewable terrain is projected behind the screen, even after panning or zooming, edge effects are removed. In this regard, it may be easier to zoom or pan, thus enabling more details associated with the scenario to be seen.

5.2 Radar terrain masking (RTM) modeling

Radar Terrain Masking (RTM) occurs when terrain obscures the view of a radar in a given direction. NRL has conducted research and developed effective models for visualizing RTM which rectify the deficiencies of current models. For example, Figure 2 represents a model for a radar taken from an operational system. It is very difficult to see the 3D extent of the radar coverage. This applies whether the model is transparent, semitransparent, or solid. Based on these findings, NRL developed a line-of-sight model [9] for RTM in the polar coordinate system with the origin at the radar site. In such a polar coordinate frame the important dimensions are range, azimuth, and elevation. By displaying two of the dimensions and iterating over the third, an animated display can be generated that (a) allows the user to see different portions of the region in isolation and (b) gives the user intuition about the overall 3D structures of the detection regions. The question that must be answered is which, if any, of the choices of coordinates to animate will provide adequate visualization of the masked regions.

Iterating on range yields an animated sequence in which the detection region starts as a point and then expands to the full radar dome. Unfortunately, this display provides little information beyond what is provided by the full radar dome alone (Figure 2). Iterating on azimuth produces a wedge [9] defined by the minimum and maximum elevation and detection range of the radar at each discretized azimuth value. The wedge sweeps around the radar position and clearly traces over the terrain masked regions. One of the principal features of the wedge display is that when combined with a static plot of a proposed strike route, a viewer can see as the wedge sweeps perpendicular to a leg of the route and determine whether the route goes through a masked region or is within the detection region of the radar. The only drawback of the wedge display is that it does not provide the viewer with a strong sense of the 3D extent of the masked regions.

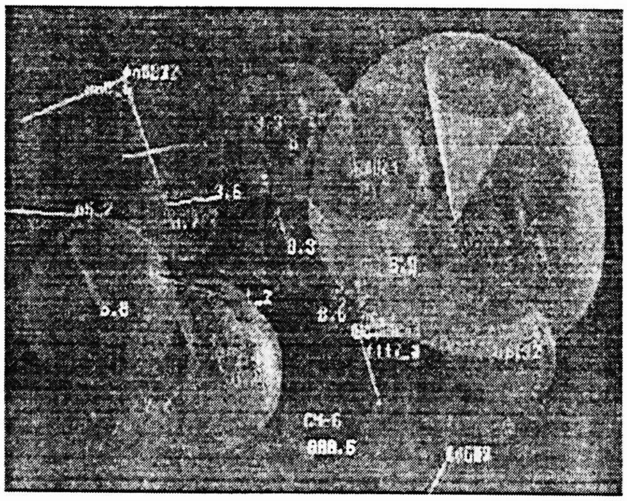

Figure 2: Semi-Transparent Radar Dome

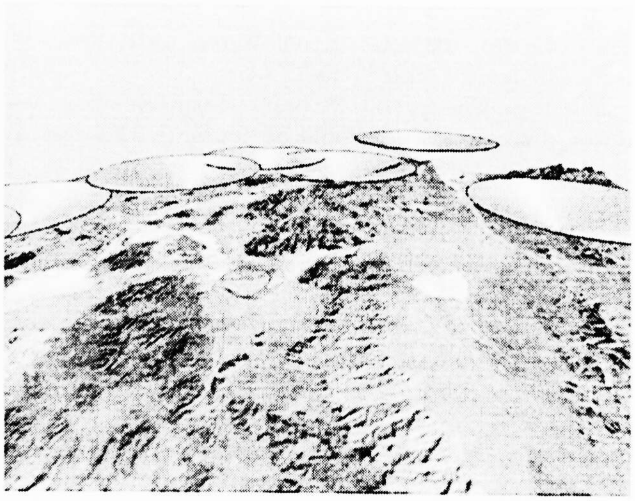
Figure 3: Cone Implementation of RTM shown at median elevation

Iterating on elevation generates a cone that continuously sweeps from minimum to maximum to minimum elevation (Figure 3). This cone appears to melt over the terrain as it sweeps down from maximum elevation onto the terrain features and, in so doing, gives a strong sense of where the masking effects are most pronounced.

In summary, by displaying individual slices of a radar detection dome in isolation, the viewer is able to discern critical information that would otherwise be occluded if all of the radar detection information were displayed at once. Furthermore, by animating a sequence of slices the viewer is able to mentally construct a complete picture of the 3D structure of the radar dome while at the same time is able to scrutinize individual slices in the animated sequence.

5.3 Advanced displays

The NRL has investigated the use of advanced displays for Command and Control (C^2) applications, particularly the use of a virtual reality responsive workbench, (Figure 4). The workbench was originally developed and built at the GMD National Research Center for Information Technology, and a copy was built at NRL for initial research. Currently, NRL is using a commercially available workbench developed by Fakespace Corporation. Whereas the original workbench top was not adjustable, the one developed by Fakespace has an adjustable table top which can tilt to approximately 45 degrees for easier viewing.

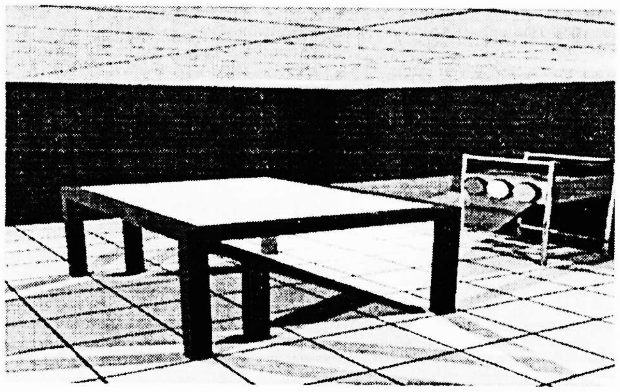

Figure 4: The Virtual Workbench

Video from the computer is sent to the projector, which projects the stereo image onto a mirror. The mirror reflects this image onto a translucent table top. Two pairs of emitters are mounted at the back two corners of the table top. The use of two pairs of emitters provides a stronger synchronization signal for the shutter glasses, however, a single emitter could be configured for use. When displaying an image on the workbench, the user can control whether the image appears to float above, or just below, the table top by adjusting the same parameter which controls where the image converges (e.g., with respect to the far/near clipping planes).

The advantage of the use of stereo on the virtual workbench arises from observing that users naturally perceive altitude in the same direction as a vector which is perpendicular to the earth. The workbench provides an environment in which users can naturally interact with objects on terrain in a natural table top environment. On a computer monitor, altitude would be in the same direction as a vector perpendicular to the computer screen. This is a little awkward to work with, especially when trying to adjust the routes in the z direction. The viewing area is also greater as compared to the standard computer monitor. This larger viewing area provides a more comfortable environment for multiple user to interact with the workbench application.

5.4 Interaction technology

The NRL has developed novel interface techniques that are well suited for interacting with the workbench application. These will be described in section 6.2.1. In the remainder of the paper we describe the STOMPM testbed, which incorporates the aforementioned technologies in autorouting, information presentation and manipulation.

6 STOMPM SYSTEM DESCRIPTION

The STOMPM systems serves as a testbed for the research and development of strike asset routing algorithms, 3D displays and interaction techniques, and research in collaborative tools and protocols. Version 1.0 of STOMPM was built mainly to support the research being conducted in the area of strike asset routing algorithms, while version 2.0 of STOMPM was built mainly to support advanced 3D displays and interaction techniques (both versions have similar models, e.g., RTM). The following sections will provide details associated with each version of STOMPM, limiting the discussion to the routing and 3D displays/interaction technologies. The STOMPM V1.0 operates in limited client/server mode while network functionality is more fully implemented in V2.0.

6.1 The STOMPM V1.0

The original STOMPM testbed (V1.0) [10] was developed in C using a free software package called FORMS [11] for the user interface. The graphics library used was the SGI GL. The main emphasis of STOMPM V1.0 is to allow the developer to easily incorporate auto-routing technology and be able to test the algorithms via a simple mouse/menu user interface. Many autorouting algorithms were implemented ranging from simple least cost path to jointly optimal routing [3]. A screenshot of the STOMPM system is shown in Figure 5. This version of STOMPM allows the user to load terrain maps (i.e., Digital Terrain Elevation Data or DTED), place assets, radar types, and targets on the terrain, specify routing parameters, and eventually choose a particular routing routine to find route(s) from the assets to the targets. The user has the ability to save/load scene files, view the environment from various pre-defined locations, get/change information about entities in the scene, and modify certain attributes associated with the visualization of this information.

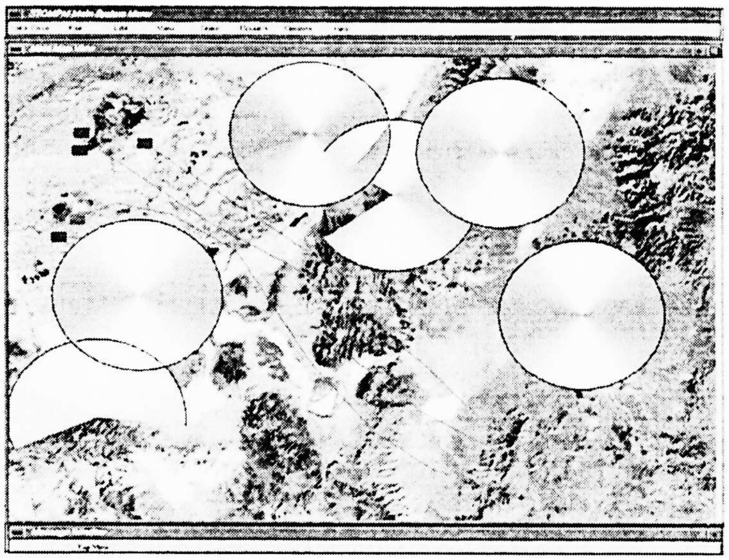

Figure 5: Screenshot of STOMPM V1.0 User interface. Shown are the RTM cones and a set of suppression and attack routes. The suppression routes open a corridor for the attack routes

Although version 1.0 of STOMPM was specifically designed to test autorouting technology, near the latter part in it's development cycle stereographics were incorporated within the interface. Since this version of STOMPM was still based on the mouse/menu style interface, it did not work too well with the workbench/stereo combination due to the fact that the menus were not drawn as a part of the environment being modeled, and only the environment and the objects contained in it were in stereo. This was part of the reason that STOMPM was eventually reimplemented. A second reason for reimplementation was due to the fact that version 1.0 was not easily extendable. Creating new objects meant defining new data structures for objects. An object oriented paradigm was investigated and eventually accepted as part of the design in order to provide a more flexible system in which new objects could be created with little effort, making use of already existing object classes.

6.2 The STOMPM V2.0

The STOMPM V2.0 is a set of object-oriented C++ toolkits that facilitate the development of virtual environment simulation & planning environments. The STOMPM toolkits as they exist now provide a means to access and use routing algorithms, visualize and interact with a scene, and collaborate with other distributed STOMPM modules. A primary goal of STOMPM was that it be useful and easily transferable to other uses or designs. Towards this end, each of the STOMPM toolkits is specific in purpose, either extending existing functionality or adding to it. Through the combined use of some or all of the toolkits, it is possible to easily build new applications and/or interfaces by adding to the existing foundation of components and coding practices. In the discussion that follows, all italicized words can be interpreted as base classes or objects derived from those base classes. In either case, the meaning of these italicized words are the same in the context in which they are used, the difference is important only in the design and implementation phase (i.e., they can be used interchangeably for the purpose of the discussion).

The purpose of STOMPM is to support the capability to generate an automated set of routes from a set of starting points (sources) to a set of sinks (targets) contained in a defined scene with obstacles. A primary goal of STOMPM V2.0 is that it continue to be an extensible application, ready for use or as the basis of new or old applications. In achieving these two goals, as with all programming, it is important to define the constraints, key dependencies, and post-conditions that define how to implement the goals more concretely. Therefore, the design of STOMPM started with the desired capabilities, which were mainly set at the beginning, but also evolved over the course of the project. Below we discuss the capabilities developed as a result of our goals.

The routing algorithms in STOMPM, given a correctly specified scene and set of routing constraints, calculates an optimal route from a source to a target. Currently there are three algorithms in V2.0, but this number will be extended in the future. There is an unconstrained router, a router that is restrained by turn-angles but trades speed for a possibly non-optimal route, and an optimal turn-angle constrained router. The algorithms are interested in those objects in the scene that represent a threat to an object attempting to get from the source to the sink. An example of such a threat is a radar, which interacts in complex ways with the surrounding terrain. Thus the router is dependent upon a certain scenario or scene the user has created, and the underlying objects that make up this scene.

STOMPM provides several different utilities that may be used to create a versatile, alterable, and extendable user interface to interact with the objects in a scene or other components of STOMPM. The first point regarding the STOMPM interface is it's ability to provide a representation of the scene which the router will use, implying both a user viewpoint, and graphic representations for all objects that make up the scene. Visualization is crucial for concepts like RTM that are most intuitively understood and correctable when visualized. Secondly, the ability to alter the user viewpoint and the objects in the scene (hereafter referred to as *SceneObjects*) is present in the form of a user *HotSpot*. The *HotSpot* is a 3D version of a mouse pointer and can be driven by a variety of input devices (mouse, 6 degree of freedom tracker, keyboard, etc.) and is used to select, move objects, alter the user's view, etc. STOMPM also is able to store and retrieve the scenes for re-use at a later time.

Though the interactions with the *HotSpot* can be varied in several ways, there is still a need for other forms of input. STOMPM also provides both keyboard support and 3D menus (which interact with the *HotSpot*). All of this assumes that the user wishes to interact directly with the scene. However, another means of changing the scene the router uses is also available within STOMPM. There exists support for distributed communication, currently limiting the users to one concurrent shared scene. Thus, it is possible to alter the layout of the scene by reading from a STOMPM feed coming from another computer over a network. The interactions with the scene are merely support to supply the router with the necessary information to do its work.

The STOMPM system is composed of a complex set of components. It is able to provide scene management, viewing, support for various input devices, a 3D menuing system (Figure 6), peer-to-peer networking support, object interaction through the *HotSpot*, provide information feedback, archivability, as well as providing hooks for other useful operations. We now describe some of the toolkits in greater detail.

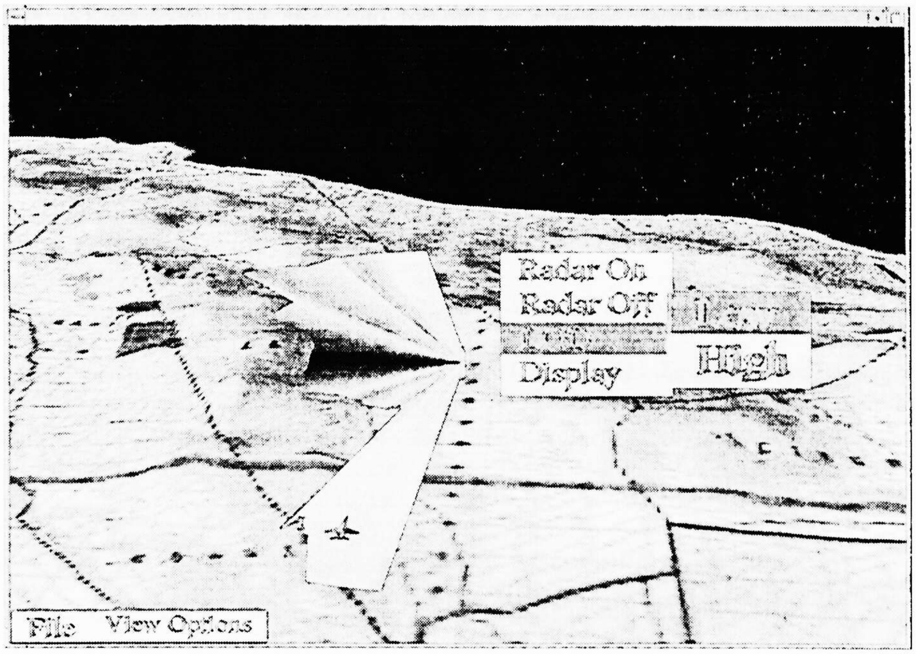

Figure 6: The STOMPM V2.0 user interface showing the HotSpot (seen as circle in "Low" per object submenu), 3D System Menu, and 3D Per/Object menus for manipulating object specific information

6.2.1 The STOMPM RTK and GATK Toolkits

STOMPM is made up of several components and toolkits. Each toolkit/component provides an independent functionality that is intrinsically different from the others. The primary toolkits that STOMPM provides are the Application Tool Kit (ATK), the Routing Tool Kit (RTK), the Graphic ATK (GATK), and the Networking TK. Within each of these toolkits are components and/or extensions to objects in other toolktis. Another crucial component is the *SceneObject* that the other toolkits either manage or use to accomplish their task. These overall toolkits provide the capabilities described in section 6.2, but are also setup to be used only in part and/or extended in part.

The ATK provides a framework basis for the other toolkits. The GATK adds graphic capabilities to the existing system. The RTK uses the ATK and *SceneObjects* to perform its routing tasks. The Networking TK extends network functionality to the ATK & GATK, but could also be considered more an extension of the existing framework than a part of the framework. The RTK and GATK will now be described in greater detail.

At the heart of the RTK are the routing algorithms. The routers use an abstract graph which represents vertices arranged in a regular 3D grid (this grid is a discretized model for the real, continuous space). Each vertex has an associated cost or danger value. Before the algorithms are called, each edge is assigned a cost based, in part, on the cost of the two vertices that it connects, as well as other factors (i.e. change in elevations, angle between the edge and the previous edge, length of the edge, exposure to threat, etc.). Starting at the source vertex, the router then visits the various vertices and sums the traversal costs. Eventually the router produces a route that minimizes total cost.

The RTK's first substantial task is to create the routing grid. The first step is to obtain the unique terrain that's present in the scene. Once the terrain is obtained, a resolution for the routing grid must be chosen. Finally, all invalid edges and vertices must be removed and the grid must be initialized. The RTK obtains the terrain by statically placing a callback with the *TerrainObject* that notifies the RTK that a terrain has been loaded.

Once the terrain has been established, the grid is created once and only once for the terrain, at a fixed resolution. Grid initialization is too costly an operation and should be done as little as possible, as a typical initialization with highly optimized code and an acceptable resolution in a typical STOMPM scenario takes as long as two minutes. Currently, the RTK can also spawn a threaded initialization routine that will allow the rest of the application to run concurrently with the grid initialization. A simple solution might be to reduce the size of the grid. The resolution of the grid, however, is a critical and difficult issue. Too coarse a grid will produce inaccurate results, while the fineness of a grid exponentially increases memory usage and initialization costs. At issue also is the type of routing that will be done, routing in a city avoiding tanks/gunmen is intrinsically different from routing to avoid radars over wide areas. They require a different fineness of grid in terms of overall distance. Thus, the user must take care to initialize the RTK with appropriate grid resolution.

Each object in the scene that has threat information that is used by the router is associated with a *ThreatObject*. The *ThreatObject* is a base class that defines the API the routers will call and use to update the routing grid. Each actual object that designates a threat will have some type of threat object that designates how cost is assigned per vertex for that object. For example, a radar (derived from *SceneObject*) would likely have a *RadarThreatObject* (derived from *ThreatObject*) associated with it. A coarse model of a *RadarObject* would specify a range and power for the radar. The *RadarThreatObject* provides the mapping function (the falloff over distance) between the vertices and the parameters inherent in the radar. The actual association of the threat information with the particular objects in question is another duty of the RTK.

The RTK cost update function is semi-complex. When each *SceneObject* has a *ThreatObject* associated with it, the router also registers all threat objects in a list of *ThreatObjects* to be updated before routing begins. Because this list can be long and the update process timely, it pays to optimize the updating. The system uses a differential update as a means of minimizing overhead. Thus, moving a threat several times will result in a subtraction volume at the original location, and an addition volume at the final location, sparing needless calculation at intermediary phases. If the objects do not move, their "update" is a null operation and trivial. Once the update list is traversed, the router is ready for routing.

We see that although the RTK has a very simple API, its duties are time-consuming, complex, and also hidden from the user. The RTK guarantees that an appropriate scale & sized grid is created, and initialized. Then it manages updating the cost of all vertices in the node in a transparent manner that is dependent upon the underlying *SceneObject* threat. Finally after all the underlying work has been done, the one function that is the point of the RTK may be called, and a viable route returned. All of this is not possible however, without the support of the ATK and *SceneObjects* that support the framework necessary for the RTK to perform its task.

We now go on to discuss the *SceneObject*, the basis of all applications purpose. The *SceneObject* represents the base class of all objects placed in the scene and therefore the object(s) which applications are interested in. All other components and parts of an application will likely use *SceneObjects* in some manner to accomplish their task. All *SceneObjects* are thought to hold in common a position, orientation, a unique ID, and the concept of being present or not in the scene. Subsequent useful information must be supplied through derivations from the base *SceneObject*.

The purpose of the GATK is to provide a base set of tools that are extendable for developing the GUI that will meet the end-users' needs. The GATK, as its name implies, inherits from the ATK, mainly because all of the components it uses must also be changed to be graphic in nature but are the same in functionality. A platform limitation of the current implementation is that the GATK uses Performer, an SGI rendering system, to do its rendering and much of its interfacing. Beyond these factors, the GATK extends the ATK to include input devices, viewing paradigms, interface mechanisms, and other IO components which are usable in part, whole, or not at all, based on user requirements.

The key extension of the GATK is the introduction of the *HotSpot*. The *HotSpot* corresponds to a 3D mouse pointer in concept. With the *HotSpot*, you can select, pop menus up, move objects, etc. It is the graphical means by which input is specified. Visually it is represented as a spot on the screen, which is simple enough to understand. The more difficult part, is how to move and position it with 6 degrees of freedom, and how it communicates with the objects it interacts with.

The *HotSpot* currently is driven by one of two input devices. The first input device is a 3-Button mouse, and the second input device is a 6 degree of freedom tracker. The tracker is currently used as a virtual pointing stick to obtain the point of interest on the screen. The point of interest for a mouse is likewise an x-y coordinate pair. The point of interest is used to project a ray from the users viewpoint to the viewing screen's position in world-space, and the first object of intersection (assuming intersection) becomes the location of the *HotSpot*. So in a very real sense, the *HotSpot* is its own device that is driven by the tracker or the mouse. Using the buttons, the tracker or mouse handler can move the *HotSpot* in or out. As the *HotSpot* is considered native to the GATK, whereas the tracker/mouse devices are not, the *HotSpot* has communication protocols established with the objects it interacts with.

Every object of significance on the display is given the ability to handle events. By this mechanism, all *GSceneObjects* (and *InterfaceObjects* such as menus) are given the ability to respond to events that relate to them. This is set up primarily for interactions with the *HotSpot*, though the developer has the option of extending this. The *HotSpot* thus at agreed times sends informational messages to the objects it interacts with. In particular, when the object is selected, unselected, hit, or unhit, the object in question is notified that the event took place. Thus, each object chooses how it will respond to particular events. Most objects will probably respond in the same way, and thus are given a default handler. For instance, when most *SceneObjects* receive a move event, they move themselves in space. However, when the *TerrainObject* receives a move event, its behavior is overridden to move the viewer of the scene thereby accomplishing the desired interaction. Thus every object has the chance to easily override behavior.

By using the *HotSpot* and its communication protocol, several features of interest have been added to STOMPM. The user can now add menus that when selected will perform developer specified callbacks. These menus are also operable on a per-object basis, radars can have one type of menu, each tank can have its own specialized menu, etc. Objects can have designated common handlers, e.g. all objects that must be placed on the ground can use one common handler that drops the object back to the ground when left in the air, while objects that can be left in the air use a different handler. An important aspect of a good user interface is the ability to move through the displayed scene quickly and easily.

STOMPM currently has two chief modes of viewing the scene. These modes are tethered viewing and egocentric viewing, which are entered into by selecting the menu items under the viewing menu. In tethered viewing mode, the user is virtually tethered to the 3D point of interest in world space, which is where the *HotSpot* was before entering viewing mode. In this case, moving to the left entails rotating about the 3D point of interest at a fixed radius in the XY plane that corresponds to the viewer's left. Viewing in egocentric mode is simpler. When the camera "moves" to the left, the viewer rotates their view and does not translate at all, effectively simulating standing on a hill top and looking around. Tethered viewing mode is of primary use when examining an object from several angles, but always looking towards the same area. Egocentric viewing mode is always looking from the same area. A third useful application of egocentric viewing is the ability to jump to any object the *HotSpot* points to, and then view from that point. Thus, it is possible to jump to a pilot's view or a view from a certain hilltop.

By using these two interface mechanisms, and other aspects of the GATK and *GSceneObject* interface it is possible to quickly maneuver through a scene and view its objects, as well as relocate them. Currently missing from the framework is an informational feedback of where the objects are as they are being moved, though this could be implemented quickly enough. Other features, such as drop-lines from the objects to give the user positioning information when they float above the terrain, bins to temporarily place objects in, means of removing all objects of a given type, etc. are all possibilities with the current STOMPM. However, as may have been apparent from the discussion there are some features that are more a part of the toolkit than others, and it is questionable where to draw the line between the toolkit components and objects proper, and those specific to the application. The end choice is always up to the developer.

7 FUTURE DIRECTION

STOMPM has the potential to grow in several areas, the foundations of which have been laid out in the toolkits mentioned. The RTK can be expanded to accommodate other types of environments or requests, such as urban, terrain-based troop routing or missile routing.

The GATK has a wealth of opportunities for expansion. It currently stores none of its parameters to file, so it is not customizable at all. There are many concerns regarding the display of other information, and options that are possible in the VR world that go beyond the components common to the X environment and personal computer operating environments. It would also be interesting to provide X widgets and menus, and/or porting to non-SGI systems, which would require a hefty rewrite of all the graphical elements.

REFERENCES

[1] Uhlmann, J.K., "A MOD LOC Vulnerability Assessment Tool", *Proceedings of the Conference of Command and Control Decision Aids*, 1992.

[2] Uhlmann, J.K., Mittu, R., "A Study to Identify Technology Shortfalls in Decision Support for Strike Warfare", *NRL Memorandum*, 1991.

[3] Zuniga, M., "The Interdependent Joint Routing Problem: Description and Algorithmic Approach"., *Proceedings of the Eleventh Annual Command and Control Decision Aids.*, Monterey, CA. 1994.

[4] Moret, B., "Network Routing Models Applied to Aircraft Routing Problems"., *Proceedings of the Winter Simulation Conference.*, 1995.

[5] Boroujerdi, A., Uhlmann, J.K., "An Efficient Algorithm for Computing Least Cost Paths with Turn Constraints," *Information Processing Letters* (in press), 1998.

[6] Boroujerdi, A., Uhlmann, J.K., "Empirical Analysis of New Methods for Computing Minimum Cost Paths with Turn Constraints," *Proceedings of the 1998 SPIE Aerosense Conference.*

[7] Tessman, Thant., "Perspectives on Stereo", IRIS Universe, *Summer 1989 issue.*

[8] Akka, Robert., "Writing Stereoscopic 3D Graphics Software using Silicon Graphics GL", *StereoGraphics Corporation Internal report*, April 17, 1991.

[9] Mittu, R., Uhlmann, J.K., "Strike Visualization in Stereo on the Virtual Workbench"., *Proceedings of the Sixth Annual Workshop on Information Technology and Systems.*, Cleveland, OH., 1996.

[10] Mittu, Ranjeev., "A Prototype System for the Evaluation of Interdependent Routing Algorithms for Military Aircraft", *Proceedings of the 11th Annual Decision Aids Conference*, Naval Postgraduate School, Monterey, CA., *Precision Strike Technology Symposium*, Johns Hopkins University Applied Physics Laboratory, Columbia, MD, 1994.

[11] Overmars, Mark H., "Forms Library: A Graphical User Interface Toolkit for Silicon Graphics Workstations"., Department of Computer Science, Utrecht University, Netherlands., 1991.